Expert Troubleshooting: Resolving MySQL Problems Quickly

Kenny Gryp <kenny.gryp@percona.com>

Netways OSDC / 2012-04-26

Resolving MySQL Problems Quickly

- The Problem
- Instrumentation
- Individual Slow Query
- Global Performance Problems
- Intermittent Performance Problems

Resolving MySQL Problems Quickly

- **The Problem**
- Instrumentation
- Individual Slow Query
- Global Performance Problems
- Intermittent Performance Problems

What is the problem?

- Find out what the real problem is!
- Is not always what others say.

Real Life Example

- Connections in CLOSE_WAIT

```
#netstat -tanp | grep 8080 | grep CLOSE_WAIT | wc
-l
82
```

- Connections to tomcat stay open!

What is the problem?

- check lsof
- Let’s reduce TIME_WAIT!

So what's the real problem?

- it's a problem with the tomcat connection pool
- connections stay open

How do you know?

- Nothing in any errorlog
- Well... no problems then...
- Where’s the data to prove it?
- ???

Getting to the problem

- “Tomcat Doesn’t Respond”
  - Almost there... How do you know it does not respond?
- “The login pages keep on loading”
  - The problem!

Things To Learn

- Find out what the real problem is.
- Looking at vague problem definitions limit your area of focus or put your in the wrong direction
- Don’t trust computers, humans and even yourself!
- Have data to back up your thoughts
- Guessing might work, but you’re lying to yourself

Resolving MySQL Problems Quickly

- The Problem
- **Instrumentation**
- Individual Slow Query
- Global Performance Problems
- Intermittent Performance Problems

MEASURING

like a boss

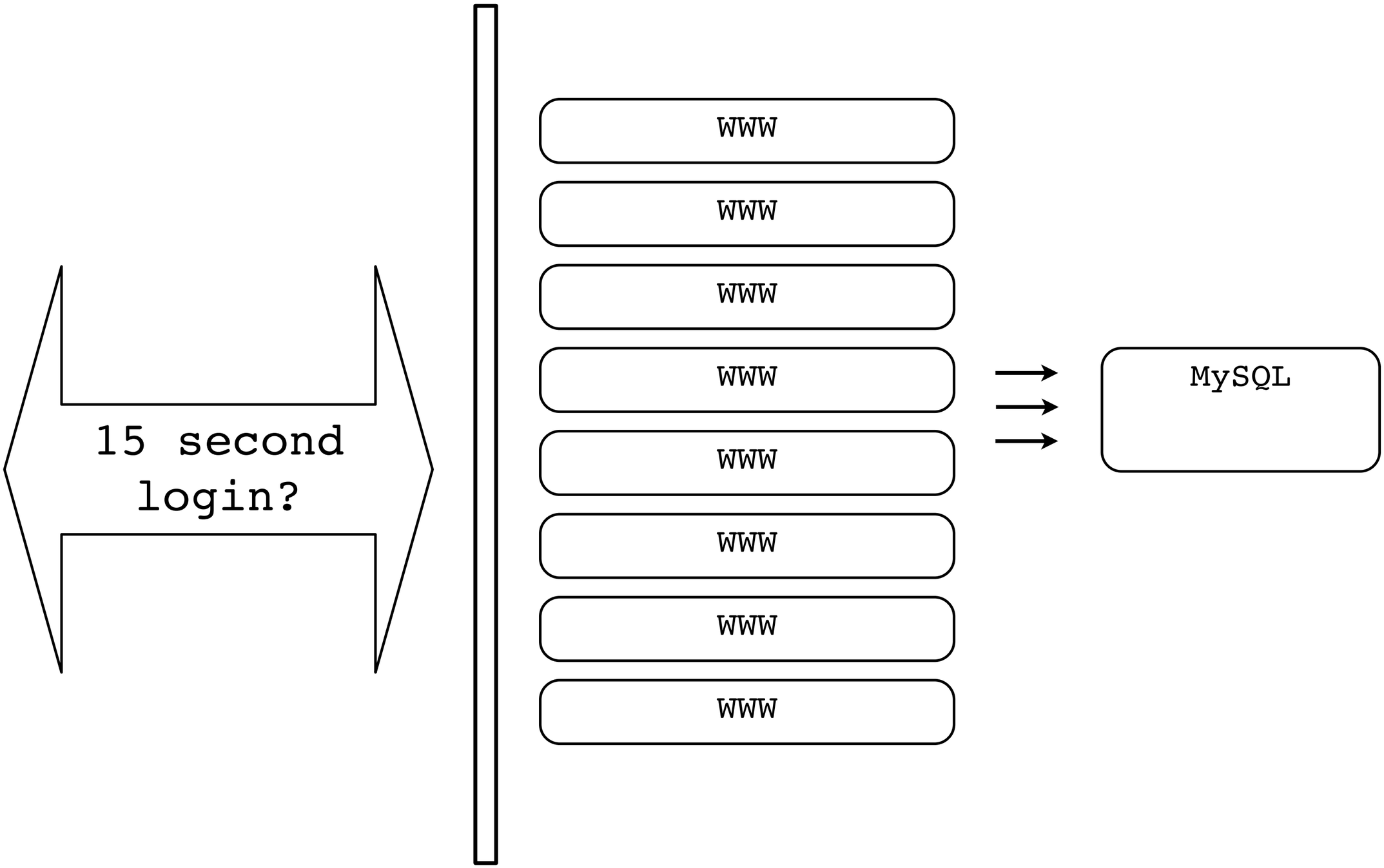
CreateFunnyPictures.com

Instrumentation

- Instrumentation is the branch of mechanical engineering that deals with measurement and control.
- "You can't control what you can't measure"
  - Tom DeMarco, *Controlling Software Projects, Management Measurement & Estimation*
- All cars give you some basic information:
  - How fast am I going?
  - How far have I gone?
  - At what rate am I consuming fuel?
  - What is the engine temperature?
  - Do I need oil?

Why do we need to instrument?
(a.k.a. “I already know where the problem is”)

Do you really?



If you said that...

- **The Database**
  - You'll be right most of the time - but you're not being 100% honest with yourself.
  - The database has more scalability challenges than the other components. For the most part we can just add web servers.

However;

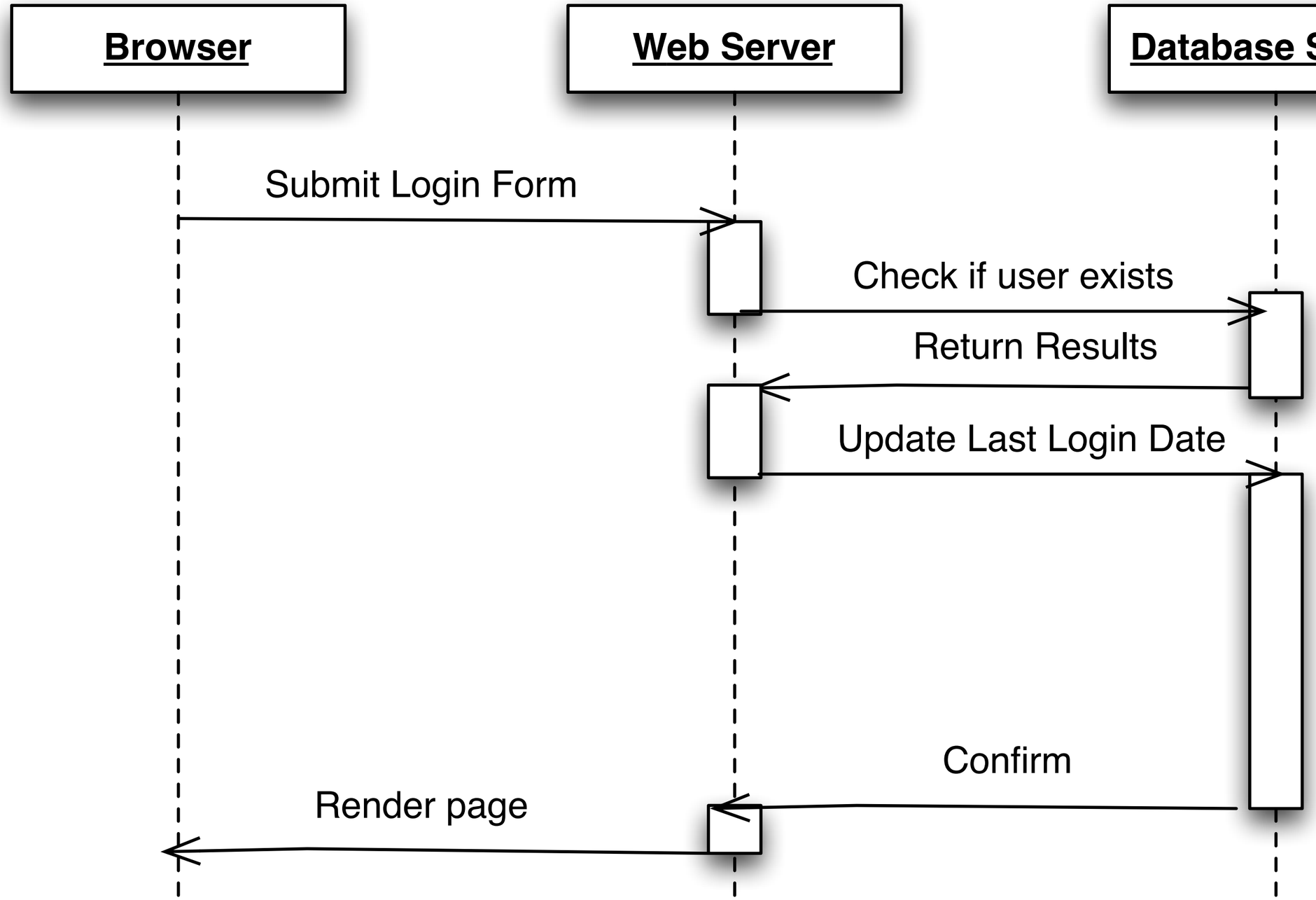
- We can lead ourselves into a real trap by guessing based on previous experience.
- Proving is probably *a lot more important* than knowing.

What's interesting...

- What's more interesting than drawing the stack is drawing **the flow of information** between each component of the stack.
- It's important to be able to do this while users execute **table7**.

Following the flow

- For a given task, measure the breakdown in time:



Wait, what.!?

- Updating the **last_login_date** takes a sizeable amount of time?
- For the value that it provides, why are we spending so long on that sub-task?

My analysis:

- **Query is:**
  - UPDATE users SET last_login_date=NOW()
    WHERE id = *N*;
- **Schema is:**
  - CREATE TABLE users (
    id INT NOT NULL PRIMARY KEY auto_increment,
    username CHAR(32) NOT NULL,
    ..
    last_login_date DATETIME,
    UNIQUE (username)
) ENGINE=MyISAM;

```
mysql> show processlist;
+------+------+-----------+------------------+---------+------+---------+-----------------------------------------------------------+
| Id   | User | Host      | db               | Command | Time | State   | Info                                                      |
+------+------+-----------+------------------+---------+------+---------+-----------------------------------------------------------+
|    1 | root | localhost | myapp_production | Query   |    0 | NULL    | show processlist                                          |
| 9688 | root | localhost | myapp_production | Query   |    2 | Updating | UPDATE users SET last_login_date=NOW() WHERE id = 1657903 |
| 9689 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 986755  |
| 9690 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 607334  |
| 9691 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1802251 |
| 9692 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1076084 |
| 9693 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 141037  |
| 9694 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1418038 |
| 9695 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1156819 |
| 9696 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 165878  |
| 9697 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1345988 |
| 9698 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1783549 |
| 9699 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 665358  |
| 9700 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 168566  |
| 9701 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1531867 |
| 9702 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 931161  |
| 9703 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 342250  |
| 9704 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 437672  |
| 9705 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 976963  |
| 9706 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 615735  |
| 9707 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1152889 |
| 9708 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1748237 |
| 9709 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 652162  |
| 9710 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1067106 |
| 9711 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1920992 |
| 9712 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1698141 |
| 9713 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1649822 |
| 9714 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 94358   |
| 9715 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 983337  |
| 9716 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1091145 |
| 9717 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 255341  |
| 9718 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 25397   |
| 9719 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1223432 |
| 9720 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1001712 |
| 9721 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1995106 |
| 9722 | root | localhost | myapp_production | Query   |    2 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 508775  |
| 9723 | root | localhost | myapp_production | Query   |    1 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 1121464 |
| 9724 | root | localhost | myapp_production | Query   |    1 | Locked  | UPDATE users SET last_login_date=NOW() WHERE id = 946613  |
```

- $ uptime
  15:00 up 11 days, 16:58, 5 users, load averages:
  0.88 0.61 0.44

- $ uptime
  15:00 up 11 days, 16:58, 5 users, load averages: 0.88 0.61 0.44

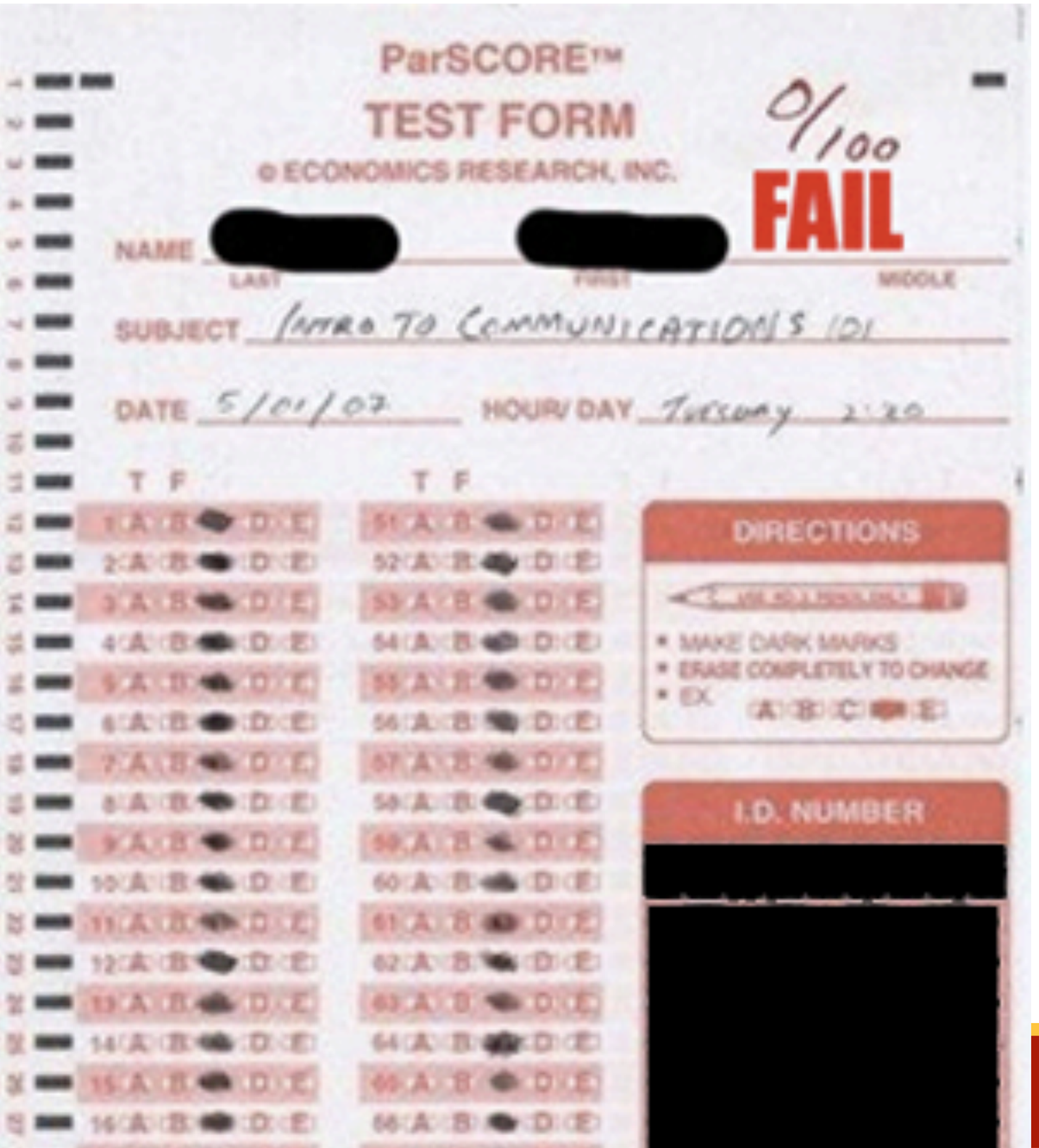
- mysql> show global status like 'slow_queries%';
  +---------------+-------+
  | Variable_name | Value |
  +---------------+-------+
  | Slow_queries | 3 |
  +---------------+-------+
  1 row in set (0.00 sec)

Guessing is misleading

- Some problems come without “load”.
  - You can’t *accurately* look at utilization rates to be able to tell where the problem is.

Guessing

- don’t just trust your guess
- try to prove it



Response from prof.

Dear [redacted]

Every year I attempt to boost my students' final grades by giving them this relatively simple exam consisting of 100 True/False questions from only 3 chapters of material. For the past 20 years that I have taught Intro Communications 101 at this institution I have never once seen someone score below a 65 on this exam. Consequently, your score of a zero is the first in history and ultimately brought the entire class average down a whole 8 points.

There were two possible answer choices: A (True) and B (False). You chose C for all 100 questions in an obvious attempt to get lucky with a least a quarter of the answers. It's as if you didn't look at a single question. Unfortunately, this brings your final

Related Concepts

Load: how much work is incoming? or, how big is the backlog?	**Utilization:** how much of a system's resources are used?	**Scalability:** what is the relationship between utilization and *R*?
Throughput: *X* - how many table7 can be done per unit of time?	**Concurrency:** how many table7 can we do at once?	**Capacity:** how big can *X* go without making other things unacceptable?

What is important...

- R = Time / Task
- X = Task / Time
- Throughput != Performance
- Is the relationship between throughput, utilization, response time and capacity.
- Queuing may occur:
  - *R* is the combination of service time and wait time.

Measuring

- Errorlog
- GLOBAL STATUS
- ENGINE INNODB STATUS
- OS metrics (memory, cpu...)
- Measure Performance
  - Response time, In application, webserver...

Example

```
* *************************** 5. row ***************************
                        ip: 91.148.82.211
                 server_ip: web08.boardreader.com
                      page: boardreader.com/s/nba29k.html?
f=47977&extended_search=1
                     utime: 0.129981
                     wtime: 0.242401
                mysql_time: 0.004417
               sphinx_time: 0.083193
        sphinx_results_time: 0.078
       mysql_count_queries: 15
             mysql_queries:
      sphinx_count_queries: 3
sphinx_real_count_queries: 3
            sphinx_queries:
                     stime: 0.008998
                    logged: 2009-07-20 20:55:48
                user_agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1;
GTB6; .NET CLR 2.0.50727; InfoPath.2)
                   referer: http://boardreader.com/fp/FileForums_14910/
PC_Games_CD_2_DVD_Conversion_47977.html
                       bot:
                 js_cookie: 1
                 page_type: search
                        id: 5ab03bc440ffa0c62610a62db988cb81
```

Available Instrumentation Tools

- Trending (Cacti graphs: http://www.percona.com/software/percona-monitoring-plugins/ and their many variants)
- APMs (Basic: http://code.google.com/p/instrumentation-for-php/),
- Write your own!

Resolving MySQL Problems Quickly

- The Problem
- Instrumentation
- **Individual Slow Query**
- Global Performance Problems
- Intermittent Performance Problems

Slow Query

- EXPLAIN
- SHOW SESSION STATUS
  - Recommended to run before and after the query.
- SHOW PROFILES
  - Available in 5.0 (limited), 5.1.
  - Breaks down the time taken on various steps of query execution.
  - Huge amount of skew in any numbers it reports under Linux.
- Slow Query Log Extended Statistics
  - Available in Percona Server
  - Will let you know examined rows, temp table on disk, sort on disk, how many IOPS in InnoDB etc.

EXPLAIN

```
mysql> EXPLAIN select STRAIGHT_JOIN count(*) as c, person_id
FROM cast_info FORCE INDEX(person_id) INNER JOIN title ON
(cast_info.movie_id=title.id) WHERE title.kind_id = 1
GROUP BY cast_info.person_id ORDER by c DESC LIMIT 1\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: cast_info
         type: index
possible_keys: NULL
          key: person_id
      key_len: 8
          ref: NULL
         rows: 8
        Extra: Using index; Using temporary; Using filesort
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: title
         type: eq_ref
possible_keys: PRIMARY,title_kind_id_exists
          key: PRIMARY
      key_len: 4
          ref: imdb.cast_info.movie_id
         rows: 1
        Extra: Using where
2 rows in set (0.00 sec)
```

SESSION STATUS

```
mysql> FLUSH STATUS
...run query....
mysql> show status like 'ha%';
+----------------------------+----------+
| Variable_name              | Value    |
+----------------------------+----------+
| Handler_commit             | 0        |
| Handler_delete             | 0        |
| Handler_discover           | 0        |
| Handler_prepare            | 0        |
| Handler_read_first         | 1        |
| Handler_read_key           | 13890229 |
| Handler_read_next          | 14286456 |
| Handler_read_prev          | 0        |
| Handler_read_rnd           | 0        |
| Handler_read_rnd_next      | 2407004  |
| Handler_rollback           | 0        |
| Handler_savepoint          | 0        |
| Handler_savepoint_rollback | 0        |
| Handler_update             | 0        |
| Handler_write              | 2407001  |
+----------------------------+----------+
15 rows in set (0.00 sec)
```

“The number of times the first entry in an index was read”

“The number of requests to read a row based on a key.”

“The number of requests to read the next row in key order.”

“The number of requests to read the next row in the data file.”

“The number of requests to insert a row in a table.”

SHOW PROFILES

- **SET profiling = 1;**
- *.. run query ..*
- **SHOW PROFILES;**
- ```
  | Query_ID | Duration     | Query
  |        1 | 211.21064300 | select STRAIGHT_JOIN
  count(*) as c, person_id  FROM cast_info FORCE
  INDEX(person_id) INNER JOIN title ON
  (cast_info.movie_id=title.id) WHERE title.kind_id =
  1  GROUP BY cast_info.person_id ORDER by c DESC
  LIMIT 1 |
  ```
- **show profile for query 1;**
  ```

SHOW PROFILES (cont.)

```
mysql> show profile for query 1;
+------------------------------+------------+
| Status                       | Duration   |
+------------------------------+------------+
| starting                     |   0.002133 |
| checking permissions         |   0.000009 |
| checking permissions         |   0.000009 |
| Opening tables               |   0.000035 |
| System lock                  |   0.000022 |
| init                         |   0.000033 |
| optimizing                   |   0.000020 |
| statistics                   |   0.000032 |
| preparing                    |   0.000031 |
| Creating tmp table           |   0.000032 |
| Sorting for group            |   0.000021 |
| executing                    |   0.000005 |
                    ..
                                                  ..
| Copying to tmp table         | 113.862209 |
| converting HEAP to MyISAM    |   0.200272 |
| Copying to tmp table on disk |  96.506704 |
| Sorting result               |   0.634087 |
| Sending data                 |   0.000047 |
| end                          |   0.000006 |
| removing tmp table           |   0.004839 |
| end                          |   0.000016 |
| query end                    |   0.000004 |
| freeing items                |   0.000064 |
| logging slow query           |   0.000004 |
| logging slow query           |   0.000003 |
| cleaning up                  |   0.000006 |
+------------------------------+------------+
25 rows in set (0.00 sec)
```

Slow Log Statistics

- SET GLOBAL long_query_time = 0;
- SET GLOBAL log_slow_verbosity = 'full';

This was executed on a machine with entirely cold caches.

```
# Time: 100924 13:58:47
# User@Host: root[root] @ localhost []
# Thread_id: 10  Schema: imdb  Last_errno: 0  Killed: 0
# Query_time: 399.563977  Lock_time: 0.000110  Rows_sent: 1  Rows_examined: 46313608
Rows_affected: 0  Rows_read: 1
# Bytes_sent: 131  Tmp_tables: 1  Tmp_disk_tables: 1  Tmp_table_sizes: 25194923
# InnoDB_trx_id: 1403
# QC_Hit: No  Full_scan: Yes  Full_join: No  Tmp_table: Yes  Tmp_table_on_disk: Yes
# Filesort: Yes  Filesort_on_disk: Yes  Merge_passes: 5
#   InnoDB_IO_r_ops: 1064749  InnoDB_IO_r_bytes: 17444847616
#   InnoDB_IO_r_wait: 26.935662
#   InnoDB_rec_lock_wait: 0.000000  InnoDB_queue_wait: 0.000000
#   InnoDB_pages_distinct: 65329
SET timestamp=1285336727;
select STRAIGHT_JOIN count(*) as c, person_id  FROM cast_info FORCE INDEX(person_id)
INNER JOIN title ON  (cast_info.movie_id=title.id) WHERE title.kind_id = 1  GROUP BY
cast_info.person_id ORDER by c DESC LIMIT 1;
```

Resolving MySQL Problems Quickly

- The Problem
- Instrumentation
- Individual Slow Query
- **Global Performance Problems**
- Intermittent Performance Problems

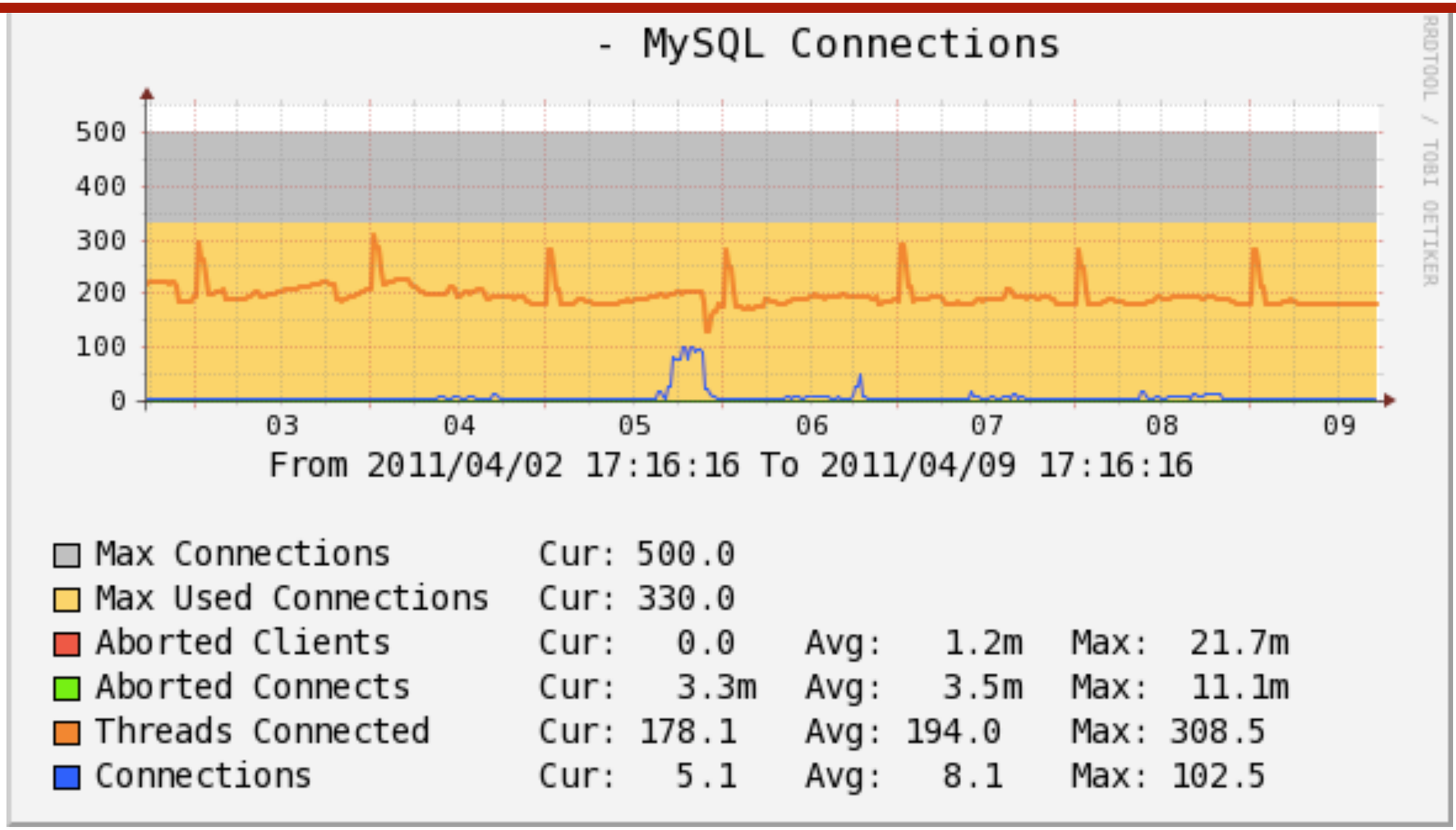
Global Performance Problems

- Example: 95% response time increased from 40ms to 200ms.
- What is going on?
  - Trending: Use graphs
    - Cacti templates (http://www.percona.com/software/percona-monitoring-plugins/) or it's variants (https://launchpad.net/percona-ganglia-mysql)
  - More granular, or ad-hoc: Look at global statistics (of db, os, hw) during the performance problems.

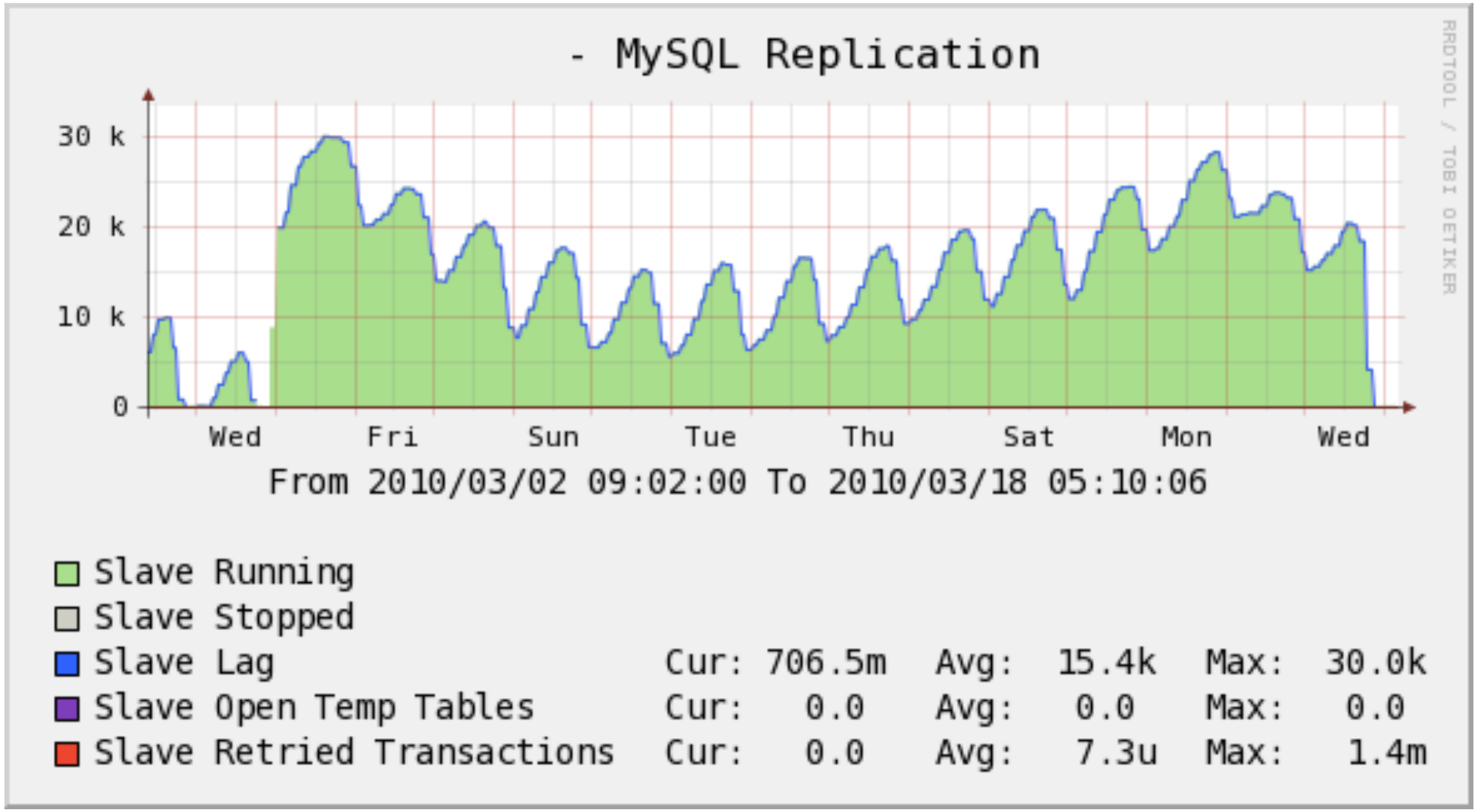
Performance Problems

- Gather information about global behavior
- Tools to use:
  - sysstat: iostat, mpstat, vmstat
  - pt-mext
  - pt-diskstats
  - pt-ioprofile
  - pt-pmp
  - pt-query-digest

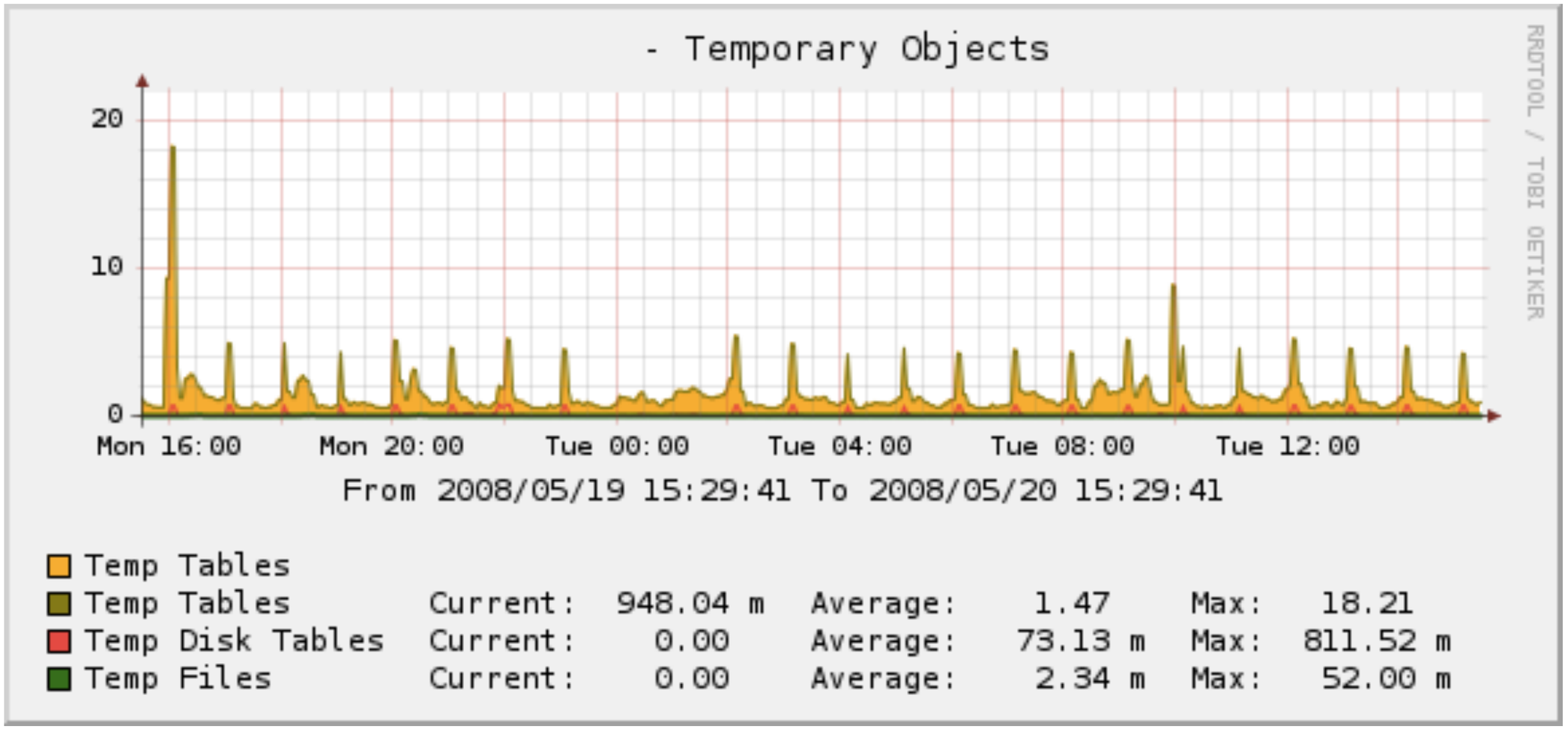
MySQL Connections

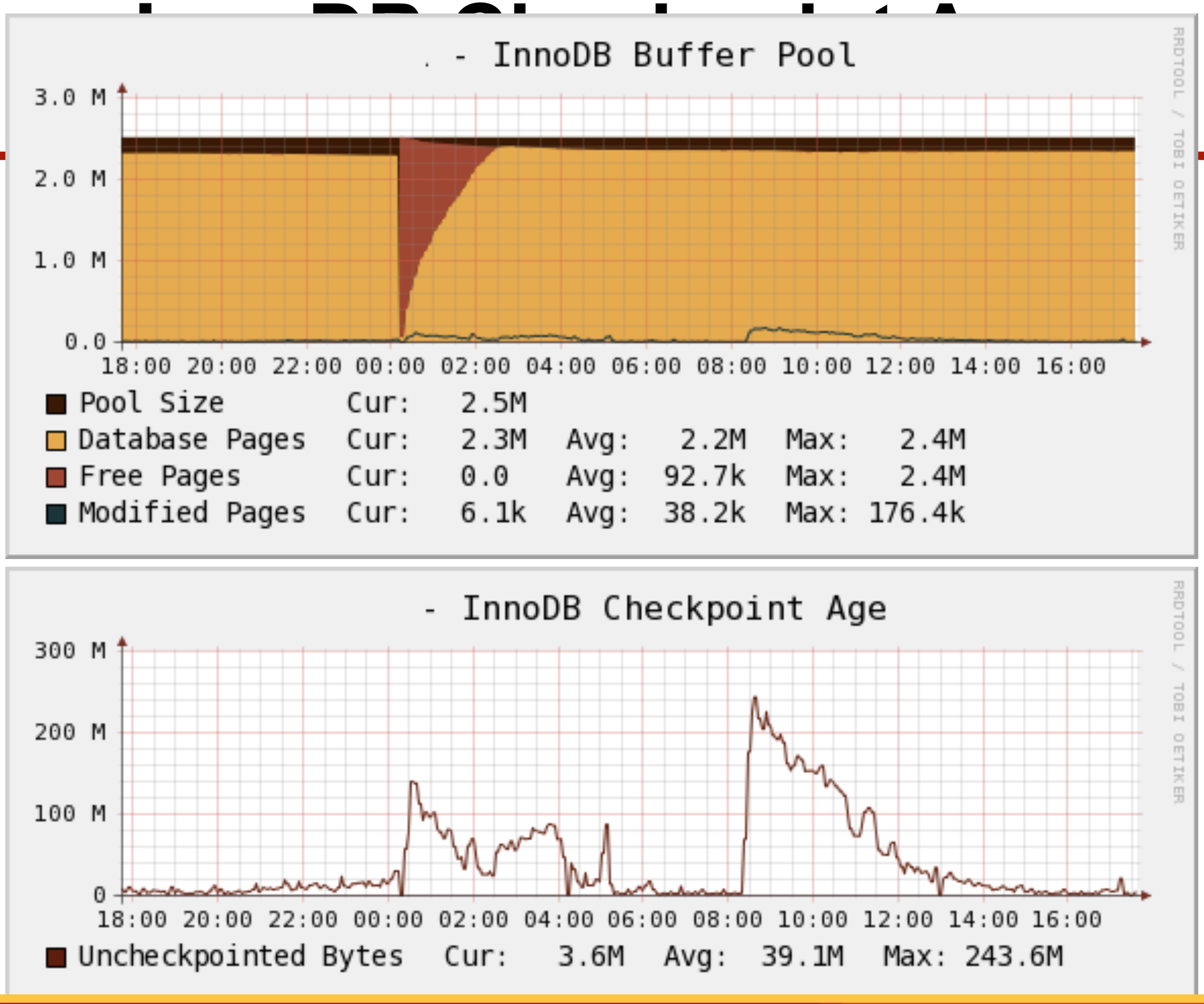


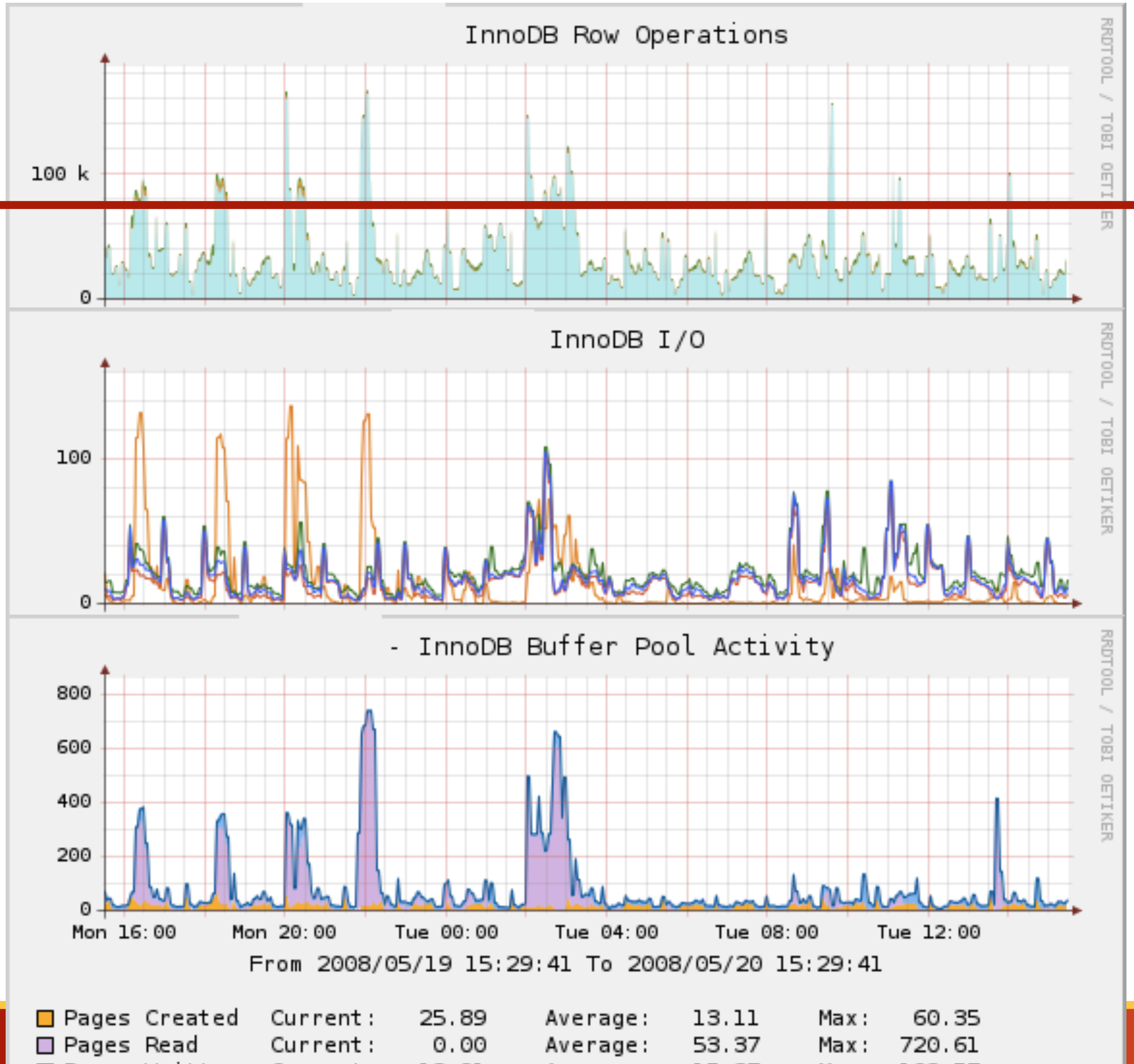
MySQL Replication



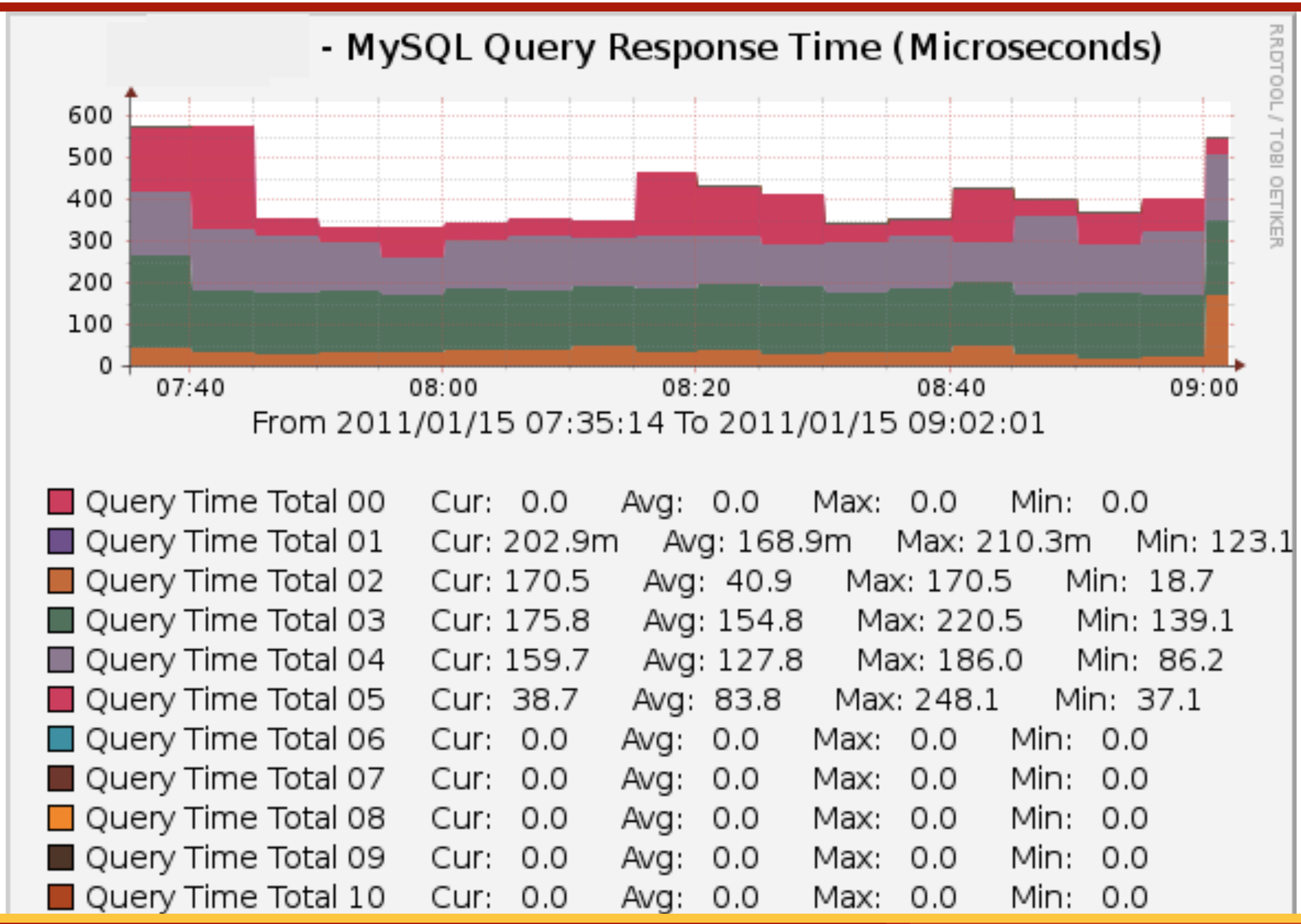
MySQL Temporary Objects



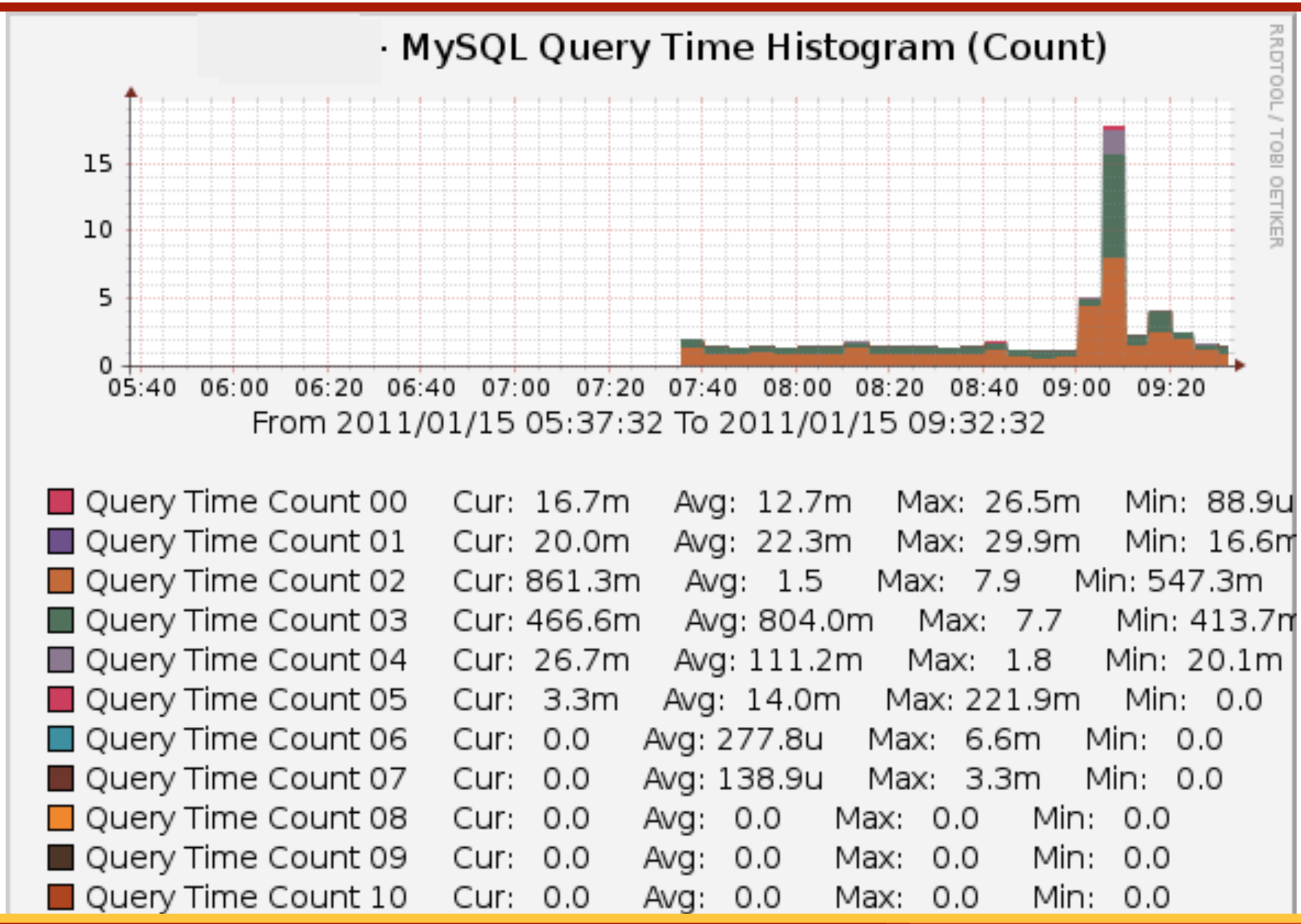

. - InnoDB Buffer Pool
3.0 M
2.0 M
1.0 M
0.0
18:00 20:00 22:00 00:00 02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00
Pool Size Cur: 2.5M
Database Pages Cur: 2.3M Avg: 2.2M Max: 2.4M
Free Pages Cur: 0.0 Avg: 92.7k Max: 2.4M
Modified Pages Cur: 6.1k Avg: 38.2k Max: 176.4k
RRDTOOL / TOBI OETIKER
- InnoDB Checkpoint Age
300 M
200 M
100 M
0
18:00 20:00 22:00 00:00 02:00 04:00 06:00 08:00 10:00 12:00 14:00 16:00
Uncheckpointed Bytes Cur: 3.6M Avg: 39.1M Max: 243.6M
RRDTOOL / TOBI OETIKER


InnoDB Row Operations
100 k
0
RRDTOOL / TOBI OETIKER
InnoDB I/O
100
0
RRDTOOL / TOBI OETIKER
- InnoDB Buffer Pool Activity
800
600
400
200
0
Mon 16:00
Mon 20:00
Tue 00:00
Tue 04:00
Tue 08:00
Tue 12:00
From 2008/05/19 15:29:41 To 2008/05/20 15:29:41
Pages Created Current: 25.89 Average: 13.11 Max: 60.35
Pages Read Current: 0.00 Average: 53.37 Max: 720.61
Pages Written Current: 12.91 Average: 15.87 Max: 102.57
RRDTOOL / TOBI OETIKER

MySQL Query Response Time


- MySQL Query Response Time (Microseconds)
RRDTOOL / TOBI OETIKER
600
500
400
300
200
100
0
07:40
08:00
08:20
08:40
09:00
From 2011/01/15 07:35:14 To 2011/01/15 09:02:01
Query Time Total 00 Cur: 0.0 Avg: 0.0 Max: 0.0 Min: 0.0
Query Time Total 01 Cur: 202.9m Avg: 168.9m Max: 210.3m Min: 123.1
Query Time Total 02 Cur: 170.5 Avg: 40.9 Max: 170.5 Min: 18.7
Query Time Total 03 Cur: 175.8 Avg: 154.8 Max: 220.5 Min: 139.1
Query Time Total 04 Cur: 159.7 Avg: 127.8 Max: 186.0 Min: 86.2
Query Time Total 05 Cur: 38.7 Avg: 83.8 Max: 248.1 Min: 37.1
Query Time Total 06 Cur: 0.0 Avg: 0.0 Max: 0.0 Min: 0.0
Query Time Total 07 Cur: 0.0 Avg: 0.0 Max: 0.0 Min: 0.0
Query Time Total 08 Cur: 0.0 Avg: 0.0 Max: 0.0 Min: 0.0
Query Time Total 09 Cur: 0.0 Avg: 0.0 Max: 0.0 Min: 0.0
Query Time Total 10 Cur: 0.0 Avg: 0.0 Max: 0.0 Min: 0.0

MySQL Query Response Time


· MySQL Query Time Histogram (Count)
RRDTOOL / TOBI OETIKER
15
10
5
0
05:40
06:00
06:20
06:40
07:00
07:20
07:40
08:00
08:20
08:40
09:00
09:20
From 2011/01/15 05:37:32 To 2011/01/15 09:32:32
Query Time Count 00 Cur: 16.7m Avg: 12.7m Max: 26.5m Min: 88.9u
Query Time Count 01 Cur: 20.0m Avg: 22.3m Max: 29.9m Min: 16.6m
Query Time Count 02 Cur: 861.3m Avg: 1.5 Max: 7.9 Min: 547.3m
Query Time Count 03 Cur: 466.6m Avg: 804.0m Max: 7.7 Min: 413.7m
Query Time Count 04 Cur: 26.7m Avg: 111.2m Max: 1.8 Min: 20.1m
Query Time Count 05 Cur: 3.3m Avg: 14.0m Max: 221.9m Min: 0.0
Query Time Count 06 Cur: 0.0 Avg: 277.8u Max: 6.6m Min: 0.0
Query Time Count 07 Cur: 0.0 Avg: 138.9u Max: 3.3m Min: 0.0
Query Time Count 08 Cur: 0.0 Avg: 0.0 Max: 0.0 Min: 0.0
Query Time Count 09 Cur: 0.0 Avg: 0.0 Max: 0.0 Min: 0.0
Query Time Count 10 Cur: 0.0 Avg: 0.0 Max: 0.0 Min: 0.0

pt-mext

```
percona@machine ~ $ ./pt-mext  -r -- mysqladmin ext -i 10 -c 3
Binlog_cache_disk_use                    0                0                0
Binlog_cache_use                         0                0                0
Bytes_received                2875788973602          1738235           346057
Bytes_sent                     863929033790           588078           536398
Com_begin                        6298644573             3516             5102
Com_delete                         23721852               26               51
Com_insert                       4454794705             1518             3287
Com_replace                       527848577              197              121
Com_select                       6993291133             8114             7594
Com_set_option                     5112076              250              262
Connections                        7331059              250              262
Created_tmp_disk_tables             113568                0                0
Created_tmp_files                     7803                0                0
Created_tmp_tables                729281259             1816              479
```

pt-mext

Handler_commit	4002481284	5295	4911
Handler_delete	7256841	10	25
Handler_discover	0	0	0
Handler_prepare	0	0	0
Handler_read_first	47274	0	0
Handler_read_key	42993091324	34920	27522
Handler_read_next	19633194815	16911	10142
Handler_read_prev	2440127	0	0
Handler_read_rnd	488760449	40	12
Handler_read_rnd_next	2731205271	268	231
Handler_rollback	5781	0	0
Handler_savepoint	0	0	0
Handler_savepoint_rollback	0	0	0
Handler_update	7022320034	10047	3329
Handler_write	7334430104	1945	3638

pt-mext

Qcache_free_blocks	2899	100	-15
Qcache_free_memory	519642808	164104	-8080
Qcache_hits	325634530	0	0
Qcache_inserts	978847229	194	104
Qcache_lowmem_prunes	19158357	0	0
Qcache_not_cached	211301010	806	798
Qcache_queries_in_cache	3677	-112	9
Qcache_total_blocks	10277	-131	6
Threads_cached	9	1	0
Threads_connected	11	-1	0
Threads_created	294	0	0
Threads_running	5	-3	0
Uptime	21912350	10	10

pt-mext

- Look at current global behavior of database
- Query Optimization necessary? (sorting_%, handler_%, range_%, tmp_table_%)
- Innodb furious flushing?
- ...

Disk Subsystem Statistics

- *iostat* commonly used:

Device:	rrqm/s	wrqm/s	r/s	w/s	rsec/s	wsec/s	avgrq-sz	avgqu-sz	await	svctm	%util
sda	0.00	73.27	0.00	54.46	0.00	1061.39	19.49	4.84	88.80	18.36	100.00
sda1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sda2	0.00	73.27	0.00	54.46	0.00	1061.39	19.49	4.84	88.80	18.36	100.00
sdb	0.00	451.49	0.99	**338.61**	7.92	6368.32	18.78	144.23	**420.93**	**2.94**	100.00
sdb1	0.00	451.49	0.99	338.61	7.92	6368.32	18.78	144.23	420.93	2.94	100.00

- %util: how many % of time at least one request was busy
- await+svctm:
  - response time
  - writes and reads combined

pt-diskstats

- reads */proc/diskstats*, shows wr/rd response time

device	rd_s	rd_avkb	rd_mb_s	rd_mrg	rd_cnc	rd_rt
sda	0.1	4.0	0.0	0%	0.0	5.0
sda2	0.1	4.0	0.0	0%	0.0	5.0
sdb	119.8	5.3	0.6	0%	0.5	4.1
sdb1	119.8	5.3	0.6	0%	0.5	4.1

pt-diskstats

-

```
device   wr_s wr_avkb wr_mb_s wr_mrg wr_cnc   wr_rt
sda      23.5    35.6     0.8    89%    1.2     5.9
sda2     23.5    35.6     0.8    89%    1.2     5.9
sdb     160.3     7.5     1.2    47%   18.3    61.0
sdb1    160.3     7.5     1.2    47%   18.3    61.0
```

pt-diskstats

-

```
device busy in_prg     io_s  qtime  stim
sda       7%      0     23.6    5.6   0.3
sda2      7%      0     23.6    5.6   0.3
sdb      47%      0    280.0   43.6   1.1
sdb1     47%      0    280.0   43.6   1.1
```

Example: Slow DROP TABLE

- Problem:
  - Database stalls when DROP TABLE is performed
- Known:
  - innodb_file_per_table=on but already using XFS (ext3 is slow in deleting files: http://www.mysqlperformanceblog.com/2009/06/16/slow-drop-table/)
  - CPU-bound

•SHOW ENGINE INNODB STATUS:

```
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 99807827, signal count 386610135
--Thread 1409755456 has waited at buf0flu.c line 1335 for 0.0000 seconds the
semaphore:
S-lock on RW-latch at 0x2aab630da3f8 '&block->lock'
a writer (thread id 1846389056) has reserved it in mode exclusive
number of readers 0, waiters flag 1, lock_word: fffffffffff00000
Last time read locked in file buf0flu.c line 1335
Last time write locked in file btr/btr0btr.c line 1447
--Thread 1462204736 has waited at row0purge.c line 665 for 52.000 seconds the
semaphore:
S-lock on RW-latch at 0x10999a0 '&dict_operation_lock'
a writer (thread id 1846389056) has reserved it in mode exclusive
number of readers 0, waiters flag 1, lock_word: 0
Last time read locked in file row0purge.c line 665
Last time write locked in file row/row0mysql.c line 3212
--Thread 1516517696 has waited at dict0boot.ic line 45 for 52.000 seconds the
semaphore:
Mutex at 0x2acae15e1d18 '&dict_sys->mutex', lock var 1
waiters flag 1
--Thread 1976047936 has waited at dict0boot.ic line 45 for 51.000 seconds the
semaphore:
Mutex at 0x2acae15e1d18 '&dict_sys->mutex', lock var 1
waiters flag 1
--Thread 1614227776 has waited at dict0boot.ic line 45 for 51.000 seconds the
```

Slow DROP TABLE

- Massive contention on an InnoDB dictionary mutex? What's going on?

Slow DROP TABLE

- pt-pmp: gdb stacktraces, count what threads are doing (http://poormansprofiler.org/)

```
# pt-pmp
66 ...,os_aio_simulated_handle,fil_aio_wait,io_handler_...
4 ...do_command,handle_one_connection
 1 select,os_thread_sleep,srv_purge_thread,start_thread,clone
 1 select,handle_connections_sockets,main
 1 ...my_net_read,cli_safe_read,read_event,handle_slave_io...
 1 do_sigwait,sigwait,signal_hand,start_thread,clone
 1 buf_LRU_invalidate_tablespace,fil_delete_tablespace,
row_drop_table_for_mysql,ha_innobase::delete_table,ha_delete_
table,mysql_rm_table_part2,mysql_rm_table,mysql_execute_comma
nd,mysql_parse,Query_log_event::do_apply_event,apply_event,ap
ply_event_and_update_pos,exec_relay_log_event,handle_slave_sq
l,start_thread,clone
```

Slow DROP TABLE

- oprofile can show use where cpu time has been spent

```
 15753796 56.0725 no-vmlinux no-vmlinux  /no-vmlinux
11834143 42.1213  mysqld     mysqld     buf_LRU_invalidate_tablespace
168823    0.6009  mysql      mysql      completion_hash_update
53667     0.1910  oprofiled  oprofiled  /usr/bin/oprofiled
42116     0.1499  mysqld     mysqld     buf_calc_page_new_checksum
32107     0.1143  mysqld     mysqld     srv_release_threads
14624     0.0521  mysqld     mysqld     srv_table_get_nth_slot
```

Slow DROP TABLE

- when innodb_file_per_table, dropping a table causes innodb to run through LRU and delete all pages in that tablespaceid
- Fix in Percona Server: innodb_lazy_drop_table=1 (http://www.mysqlperformanceblog.com/2011/04/20/drop-table-performance/), also fixed in 5.5.20.

Slow DROP TABLE

- However, another customer still had problems
- using *pt-mext*, you could see what was happening:

```
Innodb_mem_adaptive_hash
16472598592       -20889600       -14925824
-15056896         -14811136       -14909440
-14876672         -14827520       -14827520
-14909440         -15089664       -14827520
-15024128         -15024128       -14958592       ...
```

- Adaptive Hash Index: was also invalidated at DROP TABLE, fixed in 5.5.23 (http://bugs.mysql.com/bug.php?id=51325)

Optimizing Queries

- Example: bad performance: response time went up
- *pt-mext* shows a large amount of *handler_read_rnd_next*, which means a lot of tablescans are being done:

Handler_read_first	47274	0	0
Handler_read_key	42993091324	34920	27522
Handler_read_next	19633194815	16911	10142
Handler_read_prev	2440127	0	0
Handler_read_rnd	488760449	40	12
Handler_read_rnd_next	2731205271	**86212518**	**65727868**

- how to find the queries that cause this?
  - slow query log

Enhanced Slow Log Statistics

- SET GLOBAL long_query_time = 0;
- SET GLOBAL log_slow_verbosity = 'full';

```
# Time: 100924 13:58:47
# User@Host: root[root] @ localhost []
# Thread_id: 10  Schema: imdb  Last_errno: 0  Killed: 0
# Query_time: 399.563977  Lock_time: 0.000110  Rows_sent: 1  Rows_examined: 46313608
Rows_affected: 0  Rows_read: 1
# Bytes_sent: 131  Tmp_tables: 1  Tmp_disk_tables: 1  Tmp_table_sizes: 25194923
# InnoDB_trx_id: 1403
# QC_Hit: No  Full_scan: Yes  Full_join: No  Tmp_table: Yes  Tmp_table_on_disk: Yes
# Filesort: Yes  Filesort_on_disk: Yes  Merge_passes: 5
#   InnoDB_IO_r_ops: 1064749  InnoDB_IO_r_bytes: 17444847616
#   InnoDB_IO_r_wait: 26.935662
#   InnoDB_rec_lock_wait: 0.000000  InnoDB_queue_wait: 0.000000
#   InnoDB_pages_distinct: 65329
SET timestamp=1285336727;
select STRAIGHT_JOIN count(*) as c, person_id  FROM cast_info FORCE INDEX(person_id)
INNER JOIN title ON  (cast_info.movie_id=title.id) WHERE title.kind_id = 1  GROUP BY
cast_info.person_id ORDER by c DESC LIMIT 1;
```

pt-query-digest

- generate reports from
  - slow query log
    `pt-query-digest /path/to/slow.log`
  - binlog files
  - processlist
  - postgresql log files
  - general log (not so useful)
  - tcpdump files that captured traffic from: mysql, memcached, http
- group-by& order-by: db/ip/host/query_time:sum/ max/min/count

pt-query-digest

- store reports in db: **`--review,--review-table8`**
- enhanced filtering capabilities

```
'$event->{fingerprint} =~ m/^select/'
'$event->{Warning_count} > 1'
'$event->{InnoDB_IO_r_ops} > 50'
'$event->{QC_hit} = "Yes"'
'$event->{Bytes} >= m/^1_048_576/'
```

pt-query-digest

```
# 834.7s user time, 9.1s system time, 302.78M rss, 392.96M vsz
# Current date: Mon Nov 29 09:47:43 2010
# Hostname: servername
# Files: STDIN
# Overall: 670.66k total, 1.73k unique, 955.33 QPS, 3.08x concurrency _____
# Time range: 2010-11-29 09:14:29.955239 to 09:26:11.979320
# Attribute          total     min     max     avg     95%  stddev  median
# ============     ======= ======= ======= ======= ======= ======= =======
# Exec time          2163s       0       3s     3ms     2ms    29ms    89us
# Rows affecte       18.58k      0     146    0.03       0    0.49       0
# Query size        121.29M       6  21.55k  189.64  363.48  328.74   97.36
# Warning coun       438.18k      0  25.60k    0.67       0  122.19       0
# Boolean:
# No good inde   0% yes,  99% no
# No index use  10% yes,  89% no
```

pt-query-digest

```
# Profile
# Rank QID     Response time Calls  R/Call  Apdx  V/M Item
# ==== ===  ================== =====    ==== ===== ====
#    1 ...  1349.6240 62.4%  11976 0.1127 1.00  0.03 SELECT table1 table9
table2 table3 table4
#    2 ...   114.9014  5.3%    437 0.2629 1.00  0.50 SELECT table5 table6
table8 table7 table6 table8 table10
#    3 ...    92.9441  4.3%    791 0.1175 1.00  0.06 SELECT table13
#    4 ...    77.5712  3.6%     43 1.8040 0.65  0.73 SELECT table11
table12 table9 table2 table14 table15 table16 table14 table17
#    5 ...    67.1673  3.1%    296 0.2269 1.00  0.17 SELECT table8 table4
table14 table8 table18
#    6 ...    49.0330  2.3%  15630 0.0031 1.00  0.00 ADMIN CONNECT
#    7 ...    43.4990  2.0%    274 0.1588 1.00  0.12 SELECT table19
#    8 ...    30.0898  1.4%    416 0.0723 1.00  0.07 SELECT table13
#    9 ...    19.6506  0.9%  13424 0.0015 1.00  0.01 UPDATE table20
```

pt-query-digest

```
# Query 1: 17.06 QPS, 1.92x concurrency, ID 0x3928FBFF36663F33 at byte
1417466467
# This item is included in the report because it matches --limit.
# Scores: Apdex = 1.00 [1.0], V/M = 0.03
# Time range: 2010-11-29 09:14:30.052415 to 09:26:11.914796
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          1   11976
# Exec time     62    1350s     25ms    395ms    113ms    219ms     54ms     91ms
# Rows affecte   0       39       0       35     0.00        0     0.32        0
# Query size    23  28.75M   2.46k   2.46k   2.46k   2.38k        0   2.38k
# Warning coun  11  51.51k       0  12.80k    4.40        0  233.99        0
# Boolean:
# No index use  99% yes,   0% no
# String:
# Databases
# Errors        none (273/99%), #1064 (1/0%)
# Hosts         172.20.101.178
# Users         dbuser
```

pt-query-digest

```
# Query_time distribution
#   1us
#  10us  ##############################################
# 100us  ###########
#   1ms  ##
#  10ms  #
# 100ms  ##############################################################
#     1s
#  10s+
# Tables
#    SHOW TABLE STATUS LIKE 'table19'\G
#    SHOW CREATE TABLE `table19`\G
# EXPLAIN /*!50100 PARTITIONS*/
SELECT user_agent_id, search_engine FROM table19 WHERE
user_agent='Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR
1.0.3705)'\G
```

pt-query-digest

```
# Item 1: 3.41 QPS, 0.97x concurrency, ID 0xABCE5AD2A2DD1BA1 at byte
288124661
# This item is included in the report because it matches --limit.
# Scores: Apdex = 0.97 [1.0], V/M = 19.02
# Query_time sparkline: | ^_____|
# Time range: 2011-04-05 16:12:13 to 16:14:45
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          0     519
# Exec time      2     148s     11us     33s    285ms     53ms      2s     26us
# Lock time      0       5ms       0    334us      9us     66us     32us       0
# Rows sent      0      41         0       1    0.08     0.99     0.27       0
# Rows examine   1    4.97M       0 445.49k    9.80k    5.73k   49.33k       0
# Rows affecte   0       2         0       1    0.00        0    0.06       0
# Rows read      1    2.01M       0 250.47k    3.96k     1.96   27.94k    0.99
# Bytes sent     0  241.20k      11    8.01k   475.89   918.49   689.98  258.32
# Merge passes   0       0         0       0       0        0        0       0
# Tmp tables     0      15         0       1    0.03        0    0.17       0
# Tmp disk tbl   0       3         0       1    0.01        0    0.08       0
# Tmp tbl size   0    4.78k        0    4.78k    9.43        0  211.60       0
```

pt-query-digest

```
# Tmp tbl size    0    4.78k        0    4.78k     9.43        0  211.60        0
# Query size      0 100.95k       19    2.71k   199.17   363.48  206.60   151.03
# InnoDB:
# IO r bytes      0        0        0        0        0        0       0        0
# IO r ops        0        0        0        0        0        0       0        0
# IO r wait       0        0        0        0        0        0       0        0
# pages distin    1   67.99k        0   10.64k    1.26k    3.88k   2.47k    31.70
# queue wait      0        0        0        0        0        0       0        0
# rec lock wai    0        0        0        0        0        0       0        0
# Boolean:
# Filesort         0% yes,  99% no
# Full scan        7% yes,  92% no
# QC Hit          78% yes,  21% no
# Tmp table        2% yes,  97% no
# Tmp table on     0% yes,  99% no
```

Resolving MySQL Problems Quickly

- The Problem
- Instrumentation
- Individual Slow Query
- Global Performance Problems
- **Intermittent Performance Problems**

Diagnosing Intermittent Problems

- How to determine when it happens
- Tools Are Essential
  - You need to measure the problem, whether you can observe it or not.
    - Even if you see the problem happen, you can't observe 45 things at once.
    - If you can't see it happen, you can still capture diagnostic data
- Percona Toolkit
  - pt-stalk
  - pt-sift

Using SHOW STATUS

```
$ mysqladmin ext -i1 | awk '
      /Queries/{q=$4-qp;qp=$4}
      /Threads_connected/{tc=$4}
      /Threads_running/{printf "%5d %5d %5d\n", q, tc, $4}'
2147483647   136     7
  798    136     7
  767    134     9
  828    134     7
  683    134     7
  784    135     7
  614    134     7
  108    134    24
  187    134    31
  179    134    28
 1179    134     7
 1151    134     7
 1240    135     7
 1000    135     7
```

Drop in Queries Per Second
Spike of Threads_running
Threads_connected doesn't change

Using The Slow Query Log

```
$ awk '/^# Time:/{print $3, $4, c;c=0}/^# User/{c++}' slow-query.log
080913 21:52:17 51
080913 21:52:18 29
080913 21:52:19 34
080913 21:52:20 33
080913 21:52:21 38
080913 21:52:22 15
080913 21:52:23 47
080913 21:52:24 96
080913 21:52:25 6
080913 21:52:26 66
080913 21:52:27 37
080913 21:52:28 59
```

Spike, followed by a drop, in queries per second

The Diagnostic Trigger

- Determine a reliable condition to trigger the tool
- Not too low!
  - You'll get false positives
- Not too high!
  - You'll miss the problem and it will hurt longer
  - You'll diagnose the wrong problem

The Threshold

- Threads_running is very good
- Threads_connected sometimes too
- Queries per second is hard to use
  - You have to compare this vs previous sample
- PROCESSLIST works sometimes
  - Too many queries with some status (grep -c)
- Text in SHOW INNODB STATUS (awk/grep)
- Other creative triggers...

What Value Should You Use?

```
$ mysqladmin ext -i1 | awk '
      /Queries/{q=$4-qp;qp=$4}
      /Threads_connected/{tc=$4}
      /Threads_running/{printf "%5d %5d %5d\n", q, tc, $4}'
2147483647   136     7
  798    136     7
  767    134     9
  828    134     7
  683    134     7
  784    135     7
  614    134     7
  108    134    24
  187    134    31
  179    134    28
 1179    134     7
 1151    134     7
 1240    135     7
 1000    135     7
```

Configuring pt-stalk

```
Threshold=100
# Collect GDB stacktraces?
collect-gdb=0

# Collect oprofile data?
collect-oprofile=0

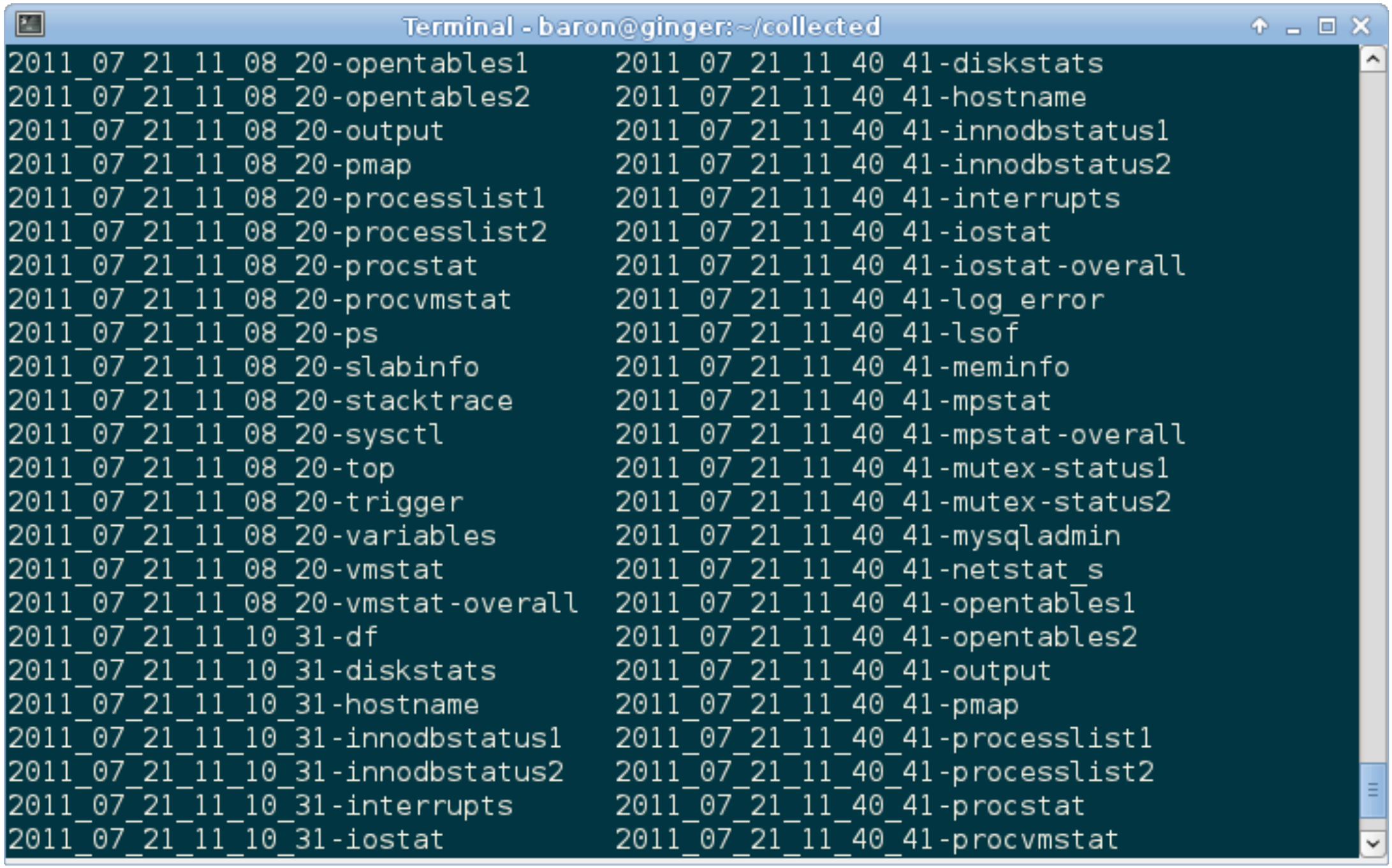
# Collect strace data?
collect-strace=0

# Collect tcpdump data?
collect-tcpdump=0
```

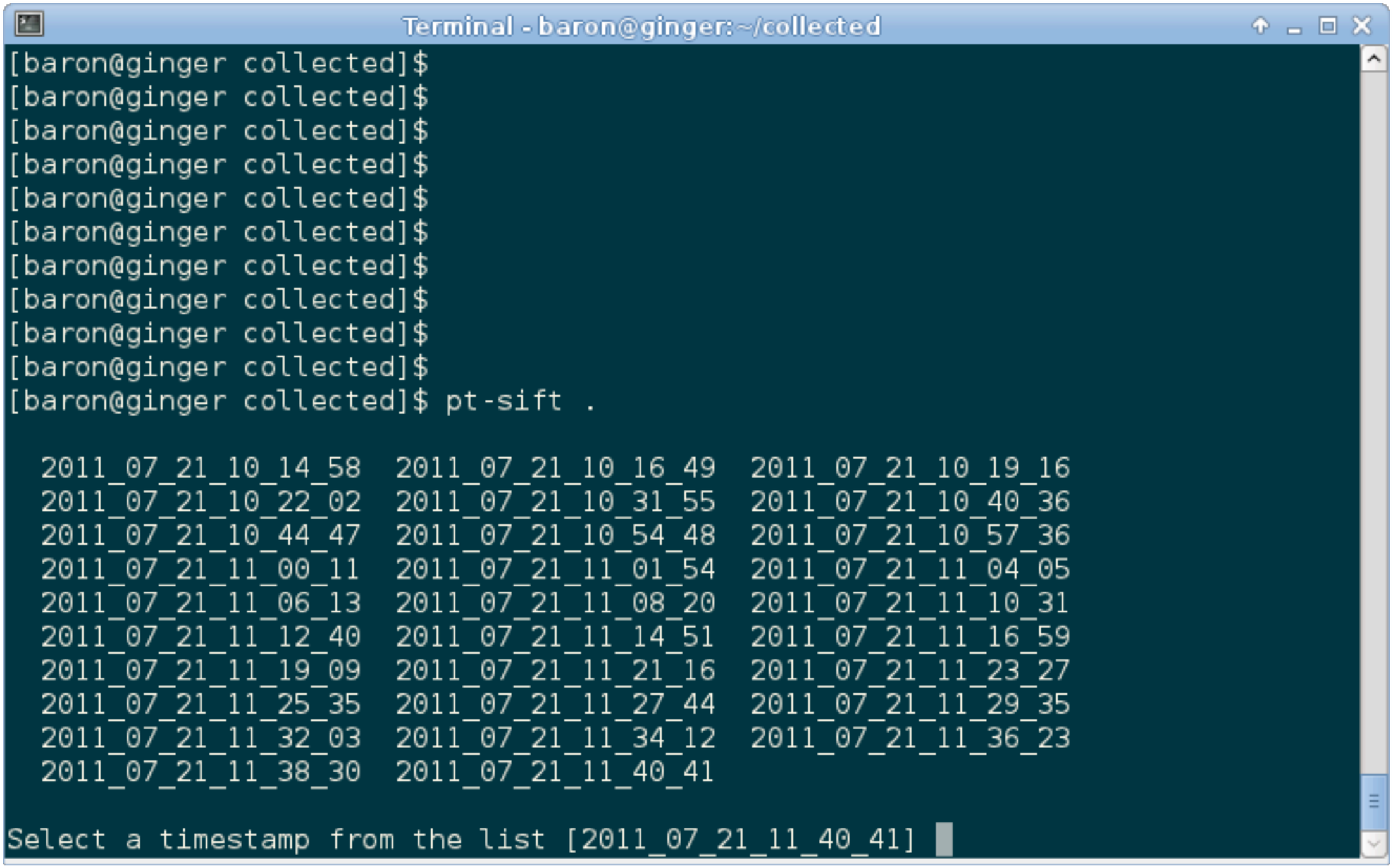
Capturing Data

- *pt-stalk* stores data in */var/lib/pt-stalk*
- There will be A LOT of data

Did I mention lots of data?



Using pt-sift



Using pt-sift

Terminal - baron@ginger:~/collected

```
--vmstat--
 r b  swpd   free   buff    cache si so bi    bo    in    cs us sy id wa st
42 0 15840 760604 142084 16869724  0  0 36   381     0     0  7  1 92  0  0
 1 0 15840 742016 142284 16911564  0  0 73 13053 31329 56995 11  3 86  0  0
wa 0% . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
--innodb--
    txns: 10xACTIVE (1s) 310xnot (0s)
    0 queries inside InnoDB, 0 queries in queue
    Main thread: flushing buffer pool pages, pending reads 1, writes 6, flush 0
    Log: lsn = 1777474532609, chkp = 1776282083670, chkp age = 1192448939
    Threads are waiting at:
    Threads are waiting on:
--processlist--
    State
    494
     19  Sending data
      5  Has sent all binlog to slave; waiting for binlog to be updated
      3  Reading from net
      2  freeing items
    Command
    496  Sleep
     25  Query
      6  Connect
      5  Binlog Dump
```

Filesystem Cache Issue

- We are seeing query pileups & high disk IO activity at random times

Filesystem Cache Issue

- Configure *pt-stalk* with *threads_running>10*

```
# grep Writeback 2011_11_03_09_44_50-meminfo
 Writeback: 13620 kB
 Writeback: 13752 kB
 Writeback: 248 kB
 Writeback: 0 kB
Writeback: 0 kB
Writeback: 200 kB
```

Filesystem Cache Issue

- After adjusting *pt-stalk* to trigger on filesystem cache writeback behavior:
  - triggers happen when binlogs get rotated:

```
-rw-r--r-- 1 root root 91 Nov 2 15:15 2011_11_02_15_15_13-trigger
-rw-r--r-- 1 root root 91 Nov 2 16:17 2011_11_02_16_17_20-trigger
-rw-r--r-- 1 root root 91 Nov 2 17:38 2011_11_02_17_38_22-trigger
...
-rw-rw---- 1 mysql mysql 1073742171 Nov 2 15:15 /db1-bin-log.003229
-rw-rw---- 1 mysql mysql 1073742976 Nov 2 16:17 /db1-bin-log.003230
-rw-rw---- 1 mysql mysql 1073742688 Nov 2 17:38 /db1-bin-log.003231
...
```

Filesystem Cache Issue

- binary logs use filesystem cache to buffer writes. (unless *sync_binlog=N*)
- When a binary log was rotated, the filesystem cache flushed the remaining part of binary log
- filesystem buffers 10% of RAM
  - set *vm.dirty_background_ratio=1*
- Reducing binary log size from 1GB to 50MB resolved the spikes.

Transaction Locking

- there is a lot of lock waits between transactions in InnoDB. Customer has no idea where it comes from, what it causes
- use Percona Server with extended slow logging
- use *pt-stalk* and trigger on:
  - transaction lock waits
  - long running transactions
  - configure to capture tcpdump data
- The tcpdump data can then be converted with *pt-query-digest*. Get the transaction session queries that causes that long running query.

Transaction Locking

```
---TRANSACTION 0 1491496991, ACTIVE 14 sec, process no 2441, OS thread
id 1253165376
15 lock struct(s), heap size 3024, undo log entries 3
MySQL thread id 3657955, query id 1020342924 falcon.website
192.168.100.8 web
Trx read view will not see trx with id >= 0 1491496992, sees < 0
1412169815
---TRANSACTION 0 1491517002, ACTIVE 2 sec, process no 2441, OS thread id
1198373184
83 lock struct(s), heap size 14320, undo log entries 138
MySQL thread id 3657956, query id 1020462952 falcon.website
192.168.100.8 web
Trx read view will not see trx with id >= 0 1491517003, sees < 0
1412169815
---TRANSACTION 0 1491525435, ACTIVE 2 sec, process no 2441, OS thread id
1189742912
75 lock struct(s), heap size 14320, undo log entries 52
MySQL thread id 3657584, query id 1020513657 eagle.website 192.168.100.7
web
```

High Transaction Locking

- Problem was application bug: a transaction was stuck in a loop
- The collected queries from that session helped development identify the problem in the application

Resolving MySQL Problems Quickly

- The Problem
  - Find out what the problem is, then look for data. Don't guess or prove your guesses
- Instrumentation
  - use trending, collect data, application performance monitoring
- Individual Slow Query
  - go beyond explain
- Global Performance Problems
  - look at global counters, statistics. Different problems require different tools
- Intermittent Performance Problems
  - *pt-stalk* on a trigger that shows the behavior change. analyze the data and adjust

Q&A

- Useful Links:

  http://www.mysqlperformanceblog.com/

  http://www.percona.com/

  http://www.percona.com/software/percona-server/

  http://www.percona.com/software/percona-toolkit/

  http://www.percona.com/software/percona-monitoring-plugins/

  <kenny.gryp@percona.com>